

**LockedMe – Virtual Key for Repositories**

The code for this project is hosted at <https://github.com/neetuverma26/LockedMe>

***The project is developed by Neetu Verma[Mpokket]***

**Project Objective:**

Completed the feathures of the project using sprints planning and then push the source code to the GitHub repository. The user interaction is via a command line only.

**Problem Statement:**

**Generic features and three operations: -**

* + Retrieving the file names in an ascending order
  + Business-level operations:
    - Option to add a user specified file to the application
    - Option to delete a user specified file from the application
    - Option to search a user specified file from the application
    - Navigation option to close the current execution context and return to the main context
  + Option to close the application

## **Sprints planning and Task completion:**

The project is planned to be completed in 4 sprint. Tasks assumed to be completed in the sprint are:

**Sprint -1**

1) Analysis and documentation for the requirements.

2) Preare flow Chart

3) Create a Welcome screen

4) create First Menu List

**Sprint 2:**

1) Call the functions on user Input

2) Function to get list of file in a predifined folder

3) Function to add File in a folder

4) function to add content in a file

**Sprint 3**

1) function to search file in the folder

2) function to delete file in the folder

3) fuction to exit from application

4) function to navigate the cutternt executuion and return to main context.

**Sprint 4**

1) Testing entire application and bug fixing

2) Preparing documentation about the application

3) creating screen shots of the application flow

**Flow Chart Of the Application:-**

[Click here to get the flow Chart](https://docs.google.com/presentation/d/1Ph-V7Vfai8w8BiW5Hlh1T37yOXMbrDXVvMLwpJuN5j8/edit?usp=sharing)

Or use below link to get the flow chart

https://docs.google.com/presentation/d/1Ph-V7Vfai8w8BiW5Hlh1T37yOXMbrDXVvMLwpJuN5j8/edit#slide=id.p

**Algorithm Of the Application:**

**Steps-1:-**

1) I use IntelliJ IDE as a editor

2) create package name as Lockedme.com

**Steps-2:-**

1) Writing a program in JAVA

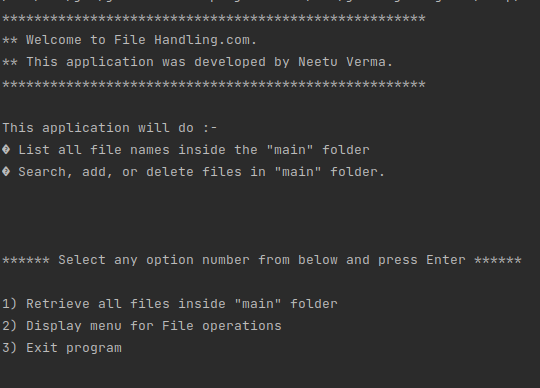
***Parent Class:-***

package Lockedme.com;  
  
import java.util.Scanner;  
  
public class Parent {  
 public static int *selectedOption*;  
 public static void main(String[] args)  
 {  
 Welcome.*welcomeScreen*();  
 MenuOptions mo=new MenuOptions();  
  
 // Scanner option1=new Scanner(System.in);  
 // System.out.println("9 � press 9 to get menu list");  
 // System.out.println("0 � press 0 to End the application");  
 // int enteredValue= option1.nextInt();  
 //selectedOption=mo.menuValue(enteredValue);  
 HandleOptions.*handleWelcomeScreenInput*();  
 }  
  
  
  
}

***Welcome Class:-***

package Lockedme.com;  
  
import java.util.Scanner;  
  
public class Welcome {  
 public static void welcomeScreen()  
 {  
 String myDetails = String.*format*("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"  
 + "\*\* Welcome to %s.com. \n" + "\*\* This application was developed by %s.\n"  
 + "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n", "File Handling", "Neetu Verma");  
 String functionsDetail = "This application will do :-\n"  
 + "� List all file names inside the \"main\" folder\n"  
 + "� Search, add, or delete files in \"main\" folder.\n";  
  
 System.*out*.println(myDetails);  
 System.*out*.println(functionsDetail);  
 }  
  
  
  
}

**We get the Output:-**



5) If User enter 1 then

***HandleOptions Class :-***

***package Lockedme.com;  
  
import java.util.List;  
import java.util.Scanner;  
  
public class HandleOptions {  
 public static void handleWelcomeScreenInput() {  
 boolean running = true;  
 Scanner sc = new Scanner(System.in);  
 do {  
 try {  
 MenuOptions mo=new MenuOptions();  
 mo.menuList();  
 int input = sc.nextInt();  
  
 switch (input) {  
 case 1:  
 FileOperations.displayAllFiles("main");  
 break;  
 case 2:  
 HandleOptions.handleFileMenuOptions();  
 break;  
 case 3:  
 System.out.println("Program exited successfully.");  
 running = false;  
 sc.close();  
 System.exit(0);  
 break;  
 default:  
 System.out.println("Please select a valid option from above.");  
 }  
 } catch (Exception e) {  
 System.out.println(e.getClass().getName());  
 handleWelcomeScreenInput();  
 }   
 } while (running == true);  
 }  
   
 public static void handleFileMenuOptions() {  
 boolean running = true;  
 Scanner sc = new Scanner(System.in);  
 do {  
 try {  
 MenuOptions.displayFileMenuOptions();  
  
 FileOperations.createMainFolderIfNotPresent("main");  
   
 int input = sc.nextInt();  
 System.out.println(input);  
 switch (input) {  
 case 1:  
 // File Add  
 System.out.println("Enter the name of the file to be added to the \"main\" folder");  
 String fileToAdd = sc.next();  
   
 FileOperations.createFile(fileToAdd, sc);  
   
 break;  
 case 2:  
 // File/Folder delete  
 System.out.println("Enter the name of the file to be deleted from \"main\" folder");  
 String fileToDelete = sc.next();  
   
 FileOperations.createMainFolderIfNotPresent("main");  
 List<String> filesToDelete = FileOperations.displayFileLocations(fileToDelete, "main");  
   
 String deletionPrompt = "\nSelect index of which file to delete?"  
 + "\n(Enter 0 if you want to delete all elements)";  
 System.out.println(deletionPrompt);  
   
 int idx = sc.nextInt();  
   
 if (idx != 0) {  
 FileOperations.deleteFileRecursively(filesToDelete.get(idx - 1));  
 } else {  
   
 // If idx == 0, delete all files displayed for the name  
 for (String path : filesToDelete) {  
 FileOperations.deleteFileRecursively(path);  
 }  
 }  
   
  
 break;  
 case 3:  
 // File/Folder Search  
 System.out.println("Enter the name of the file to be searched from \"main\" folder");  
 String fileName = sc.next();  
   
 FileOperations.createMainFolderIfNotPresent("main");  
 FileOperations.displayFileLocations(fileName, "main");  
  
   
 break;  
 case 4:  
 // Go to Previous menu  
 return;  
 case 5:  
 // Exit  
 System.out.println("Program exited successfully.");  
 running = false;  
 sc.close();  
 System.exit(0);  
 default:  
 System.out.println("Please select a valid option from above.");  
 }  
 } catch (Exception e) {  
 System.out.println(e.getClass().getName());  
 handleFileMenuOptions();  
 }  
 } while (running == true);  
 }  
}***

***MenuOptions Class:-***

***package Lockedme.com;  
  
  
import java.util.Scanner;  
  
public class MenuOptions implements Menu{  
 private static int newOption;  
 public void menuList()  
 {  
 String menu = "\n\n\*\*\*\*\*\* Select any option number from below and press Enter \*\*\*\*\*\*\n\n"  
 + "1) Retrieve all files inside \"main\" folder\n" + "2) Display menu for File operations\n"  
 + "3) Exit program\n";  
 System.out.println(menu);  
 }  
  
  
 public static void displayFileMenuOptions() {  
 String fileMenu = "\n\n\*\*\*\*\*\* Select any option number from below and press Enter \*\*\*\*\*\*\n\n"  
 + "1) Add a file to \"main\" folder\n" + "2) Delete a file from \"main\" folder\n"  
 + "3) Search for a file from \"main\" folder\n" + "4) Show Previous Menu\n" + "5) Exit program\n";  
  
 System.out.println(fileMenu);  
 }  
 public void endApplication()  
 {  
 System.out.println("Application is ended. Re-run to start again.");  
 }  
 @Override  
 public int menuValue(int a)  
 {  
 return a;  
 }  
 @Override  
 public void firstMenu(int optionId)  
 {  
 Scanner option1=new Scanner(System.in);  
 int enteredValue= option1.nextInt();  
 if(optionId==9)  
 {  
 this.menuList();  
 }  
 if(optionId==0)  
 {  
 this.endApplication();  
 }  
 if(optionId==1)  
 {  
 HandleOptions.handleWelcomeScreenInput();  
 }  
  
 }  
}***

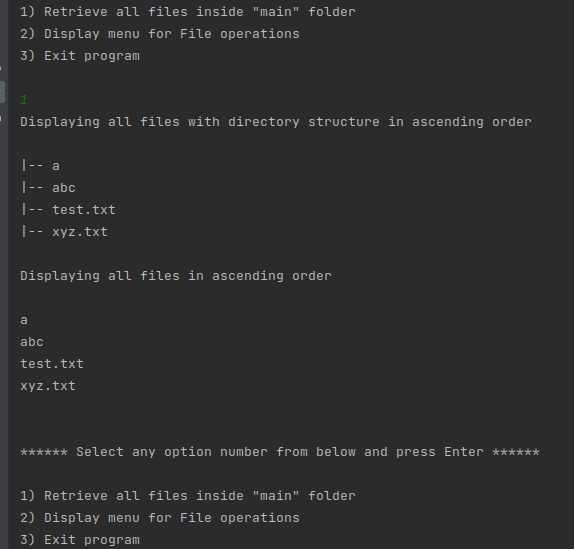
***Interface Menu:-***

***package Lockedme.com;  
  
public interface Menu {  
 public int menuValue(int option);  
 public void firstMenu(int OptionId);  
 public void menuList();  
 public void endApplication();  
}***

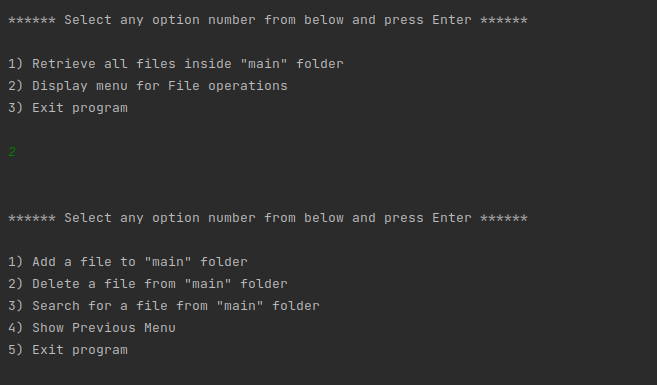
***FileOperations Class:-***

***package Lockedme.com;  
  
import java.io.File;  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.Path;  
import java.nio.file.Paths;  
import java.util.ArrayList;  
import java.util.Arrays;  
import java.util.Collections;  
import java.util.List;  
import java.util.Scanner;  
import java.util.stream.Collectors;  
import java.util.stream.IntStream;  
  
public class FileOperations {  
  
 public static void createMainFolderIfNotPresent(String folderName) {  
 File file = new File(folderName);  
  
 // If file doesn't exist, create the main folder  
 if (!file.exists()) {  
 file.mkdirs();  
 }  
 }  
  
 public static void displayAllFiles(String path) {  
 FileOperations.createMainFolderIfNotPresent("main");  
 // All required files and folders inside "main" folder relative to current  
 // folder  
 System.out.println("Displaying all files with directory structure in ascending order\n");  
  
 // listFilesInDirectory displays files along with folder structure  
 List<String> filesListNames = FileOperations.listFilesInDirectory(path, 0, new ArrayList<String>());  
  
 System.out.println("Displaying all files in ascending order\n");  
 Collections.sort(filesListNames);  
  
 filesListNames.stream().forEach(System.out::println);  
 }  
  
 public static List<String> listFilesInDirectory(String path, int indentationCount, List<String> fileListNames) {  
 File dir = new File(path);  
 File[] files = dir.listFiles();  
 List<File> filesList = Arrays.asList(files);  
  
 Collections.sort(filesList);  
  
 if (files != null && files.length > 0) {  
 for (File file : filesList) {  
  
 System.out.print(" ".repeat(indentationCount \* 2));  
  
 if (file.isDirectory()) {  
 System.out.println("`-- " + file.getName());  
  
 // Recursively indent and display the files  
 fileListNames.add(file.getName());  
 listFilesInDirectory(file.getAbsolutePath(), indentationCount + 1, fileListNames);  
 } else {  
 System.out.println("|-- " + file.getName());  
 fileListNames.add(file.getName());  
 }  
 }  
 } else {  
 System.out.print(" ".repeat(indentationCount \* 2));  
 System.out.println("|-- Empty Directory");  
 }  
 System.out.println();  
 return fileListNames;  
 }  
  
 public static void createFile(String fileToAdd, Scanner sc) {  
 FileOperations.createMainFolderIfNotPresent("main");  
 Path pathToFile = Paths.get("./main/" + fileToAdd);  
 try {  
 Files.createDirectories(pathToFile.getParent());  
 Files.createFile(pathToFile);  
 System.out.println(fileToAdd + " created successfully");  
  
 System.out.println("Would you like to add some content to the file? (Y/N)");  
 String choice = sc.next().toLowerCase();  
  
 sc.nextLine();  
 if (choice.equals("y")) {  
 System.out.println("\n\nInput content and press enter\n");  
 String content = sc.nextLine();  
 Files.write(pathToFile, content.getBytes());  
 System.out.println("\nContent written to file " + fileToAdd);  
 System.out.println("Content can be read using Notepad or Notepad++");  
 }  
  
 } catch (IOException e) {  
 System.out.println("Failed to create file " + fileToAdd);  
 System.out.println(e.getClass().getName());  
 }  
 }  
  
 public static List<String> displayFileLocations(String fileName, String path) {  
 List<String> fileListNames = new ArrayList<>();  
 FileOperations.searchFileRecursively(path, fileName, fileListNames);  
  
 if (fileListNames.isEmpty()) {  
 System.out.println("\n\n\*\*\*\*\* Couldn't find any file with given file name \"" + fileName + "\" \*\*\*\*\*\n\n");  
 } else {  
 System.out.println("\n\nFound file at below location(s):");  
  
 List<String> files = IntStream.range(0, fileListNames.size())  
 .mapToObj(index -> (index + 1) + ": " + fileListNames.get(index)).collect(Collectors.toList());  
  
 files.forEach(System.out::println);  
 }  
  
 return fileListNames;  
 }  
  
 public static void searchFileRecursively(String path, String fileName, List<String> fileListNames) {  
 File dir = new File(path);  
 File[] files = dir.listFiles();  
 List<File> filesList = Arrays.asList(files);  
  
 if (files != null && files.length > 0) {  
 for (File file : filesList) {  
  
 if (file.getName().startsWith(fileName)) {  
 fileListNames.add(file.getAbsolutePath());  
 }  
  
 // Need to search in directories separately to ensure all files of required  
 // fileName are searched  
 if (file.isDirectory()) {  
 searchFileRecursively(file.getAbsolutePath(), fileName, fileListNames);  
 }  
 }  
 }  
 }  
  
 public static void deleteFileRecursively(String path) {  
  
 File currFile = new File(path);  
 File[] files = currFile.listFiles();  
  
 if (files != null && files.length > 0) {  
 for (File file : files) {  
  
 String fileName = file.getName() + " at " + file.getParent();  
 if (file.isDirectory()) {  
 deleteFileRecursively(file.getAbsolutePath());  
 }  
  
 if (file.delete()) {  
 System.out.println(fileName + " deleted successfully");  
 } else {  
 System.out.println("Failed to delete " + fileName);  
 }  
 }  
 }  
  
 String currFileName = currFile.getName() + " at " + currFile.getParent();  
 if (currFile.delete()) {  
 System.out.println(currFileName + " deleted successfully");  
 } else {  
 System.out.println("Failed to delete " + currFileName);  
 }  
 }  
}***

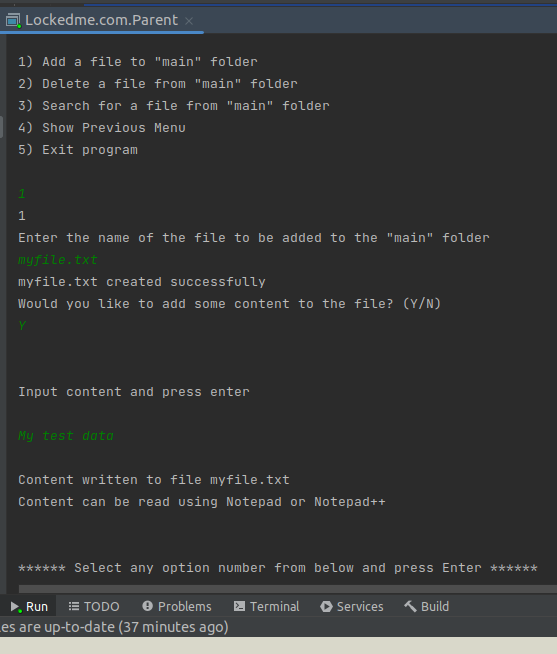
***We Get below Outputs by entering menu options:-***



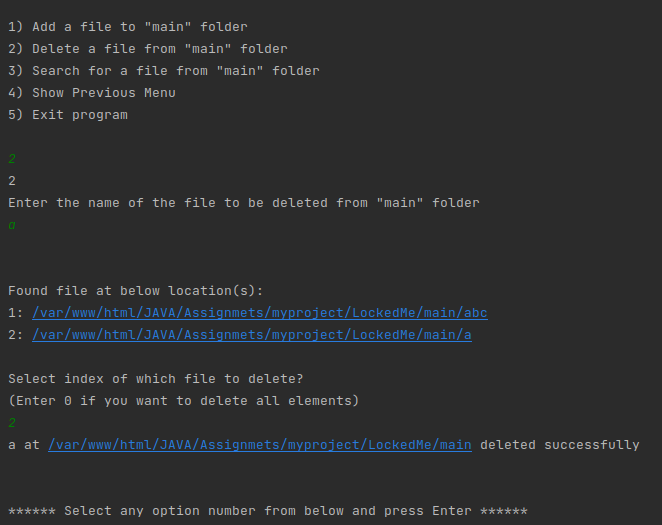
6) If user enter 2 then we get:-



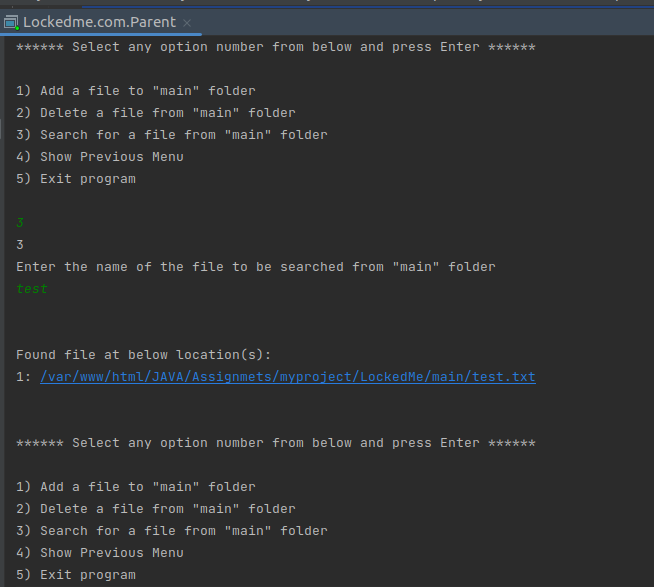
7) If user enter 1 then we get



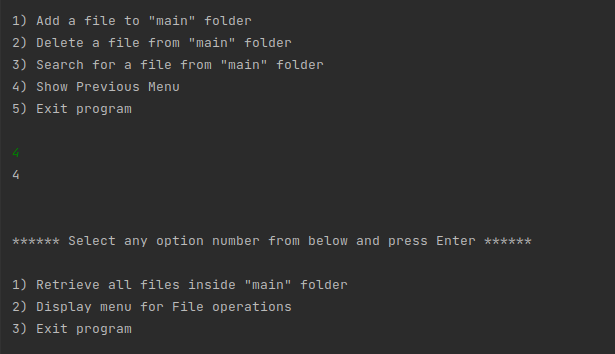
8) If user enter 2 then we get



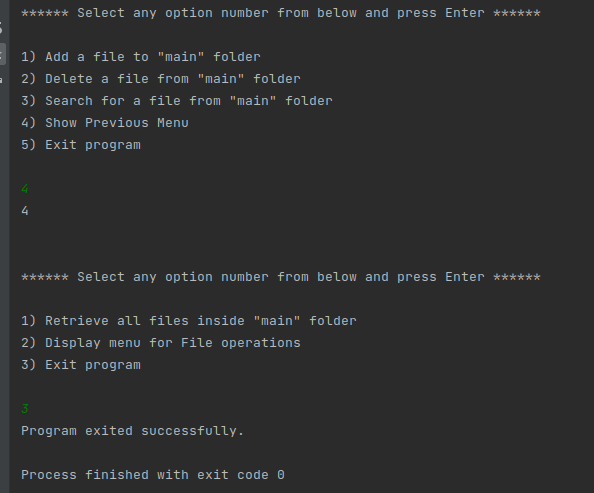
9) If user enter 3 then we get :-



10) If user enter 4 then we get:-



11) If user enter 3 or 5 then we get



12) If user enter any other number not listed in current menu then we get

